RDMAttack! or How I Learned to Stop Worrying and Authenticate at Line Rate

Stewart Grant, Shu-Ting Wang, Yibo Guo
University of California San Diego

1 Abstract

RDMA over converged ethernet (RoCEv2) is a high performance network protocol for directly accessing the memory of a remote machine by bypassing its CPU. The protocol's design prioritizes performance above all else, leaving security properties such as authenticity and confidentiality by the wayside. Security is largely unnecessary in the intimately coupled compute clusters RoCEv2 was designed for. RDMA's performance benifits have been widely recognized, and it has been integrated into a wide set of dataceter applications. Many of these general purpose applications run on diverse sets of machines (sometimes across data centers) which has propelled RDMA into high risk environments it was never designed for.

In this work we consider the design of secure RDMA. As motivation for the redesign, we hijack RDMA by performing a trivial man in the middle attack. We show that an unsophisticated attacker in control of a switch's routing table can gain full control over plaintext RDMA payloads, reading and writing to arbitrary victim addresses. Securing RDMA requires careful consideration of its performance; practitioners will not adopt a solution which degrades their high performance applications. We benchmark common encryption algorithms and demonstrate that commodity CPU's cannot encrypt at the beefy bandwidth's (100 and 400Gbps) RDMA is designed for. Our conclusion is that RDMA encryption must be implemented as a NIC offloaded function to achieve line rate performance.

2 Introduction

In ultra high performance distributed applications servicing requests for data with a CPU is a faux pas. The latency incurred by having a CPU service a network interrupt, copy memory to a buffer then to the Network Interface Card (NIC), and trigger a send is considered unseemly. To save face protocols for bypassing the CPU which allow Remote Direct Memory Access (RDMA) by NIC's are supported by most well-to-do operating systems. RDMA over converged ethernet is an en vogue instantiation of RDMA among elite data centre practitioners. In the high performance upper crust of data centre clusters security is an after thought as their traffic is largely partitioned from riff-raff applications run by potentially malicious individuals. As such RoCEv2 was designed as a separate class of protocol with minimal security considerations.

RoCE packets are bare unencrypted bytes, easily voyeurable by an attacker. The scant protection provided by the protocol comes as a checksum (iCRC) and 16 bit partition identification key used by the IOMMU to reject accesses to memory addresses not registered for RDMA. These minimal mechanisms for integrity and protection leave RoCE exposed to attackers. A man in the the middle can peek into every read revealing potentially sensitive program state, or simply modify the packet entirely reading or writing to arbitrary positions in a victims address space.

Securing RoCE, due to its performance aims, requires care. Network line rate rages towards terrabits per seconds, and RDMA is expected to scale in order to saturate links. Any encryption used to secure the protocol must operate at a competitive rate or risk becoming a latency bottleneck. Additionally part's of RoCE protocol are offloaded to the NIC itself, meaning that any encryption performed by the CPU must be carefully coordinated between the two asynchronous hardware devices.

In this work we Attack RDMA; first we show how a trivial man in the middle attack can be performed on the protocol by any attacker able to own a switch between two RDMA enabled hosts. We demonstrate the power of the attack by demonstrating full control over the protocol by an attacker: the ability to observer all traffic, modify payloads allowing reads and writes to arbitrary locations within an RDMA enabled region. We also demonstrate that an attacker can steer RDMA traffic to perform throwhammer attacks. Post attack we discuss the design space of a solution for secure RDMA, we contrast the bandwidths attainable by a variety of CPU encryption al-

gorithms and NIC offloaded encryption. Our finding lead us to believe that any standard secure RDMA algorithm must be implemented as an offloaded function to the NIC itself, in order not to interfere with 400, and potential terabit network bandwidths.

3 Background

3.1 RDMA

Remote Direct Memory Access, or RDMA, is a low-latency, high-throughput networking technology which enables direct memory access (DMA) in the form of read-/write requests which bypass the CPU of the remote machine.

For performance RDMA utilizes zero-copy from applications to the NIC which remove the latency of copying data from user to kernel space with a CPU. This approach bypasses the kernel which further reduces context switch latency. Memory accesses on a remote machine are managed entirely between the NIC and main memory which eliminate all CPU and kernel induced overhead. These performance enhancements allow RDMA to achieve μs-level latency and 100/200 Gbps throughput in modern RDMA-enabled NICs (RNICs).

3.2 RoCE and iWARP

RDMA over Converged Ethernet (RoCE) is a popular implementation of RDMA. RoCE was originally designed to run over InfiniBand network and uses InfiniBand-specific headers for routing.

RoCE in it's initial instantiation (RoCEV1) was a link later protocol. RoCEV1 packet headers are placed directly after an Ethernet header, see 1. RoCE practitioners craved routability and in it's second incarnation RoCEV2 InfiniBand headers were placed after both an IP and UDP header. RoCEV2 (sometimes referred to as routing RoCE (RRocE) increased the protocols scope and scalability without considering the security implications of enabling the protocol to route over the internet.

InfiniBand networks are designed to be lossless. To prevent RDMA packets from being dropped or lost InfiniBand uses credit-based flow control to prioritize RDMA payloads. By extension all RoCEV2 networks are expected to be lossless, a single lost RDMA packet causes catastrophic performance degradation in RoCEV2. To create the illusion of a lossless network outside of the InfiniBand environment datacenter practitioners use the standardized 802.1 Qbb Priority-based Flow Control, or PFC [1].

RoCE has comparable raw performance characteristics with RDMA on InfiniBand network. It has made it possible to run RDMA in datacenters and it's gaining popularity in recent years. Cloud operators like Microsoft Azure have started providing RDMA-capable virtual machines [2].

iWARP is an alliterative implementation of RDMA to RoCE. It drops the necessity of a lossless networking, and instead sends RDMA packets over a traditional TCP connection. While iWARP eliminates the need for hardware supported priority flow control it takes a drastic hit in performance by incurring the latency of a TCP stack. iWARPs poor performance in comparison to RoCE has lead to low adoption among high performance application designers.

3.3 RDMA Security

RDMA protocol was originally designed to operate on High Performance Computing (HPC) clusters. Such clusters are typically bespoke for a particular applications such as large-scale physical simulations, and are isolated from other networks inside a separated and trusted facility which have low security requirements.

The recent adoptions of RDMA in datacenters for cloud computing undermines the assumption of separation and trust. Operating RDMA for cloud environment assumes that malicious users might be able to perform side-channel attacks or even compromise other virtual machines simply by colocation. Concrete defense measures need to be addressed to counter the emerging security issues from datacenter-wide RDMA adoptions.

RDMA transfers data in raw bytes in favor of performance. This design decision is only acceptable in the context of HPC clusters, not in a general datacenter . Transferring encrypted data is an important but minimal requirement for securing RDMA. Protocol security is a holistic task encompassing every level of the network stack. For instance both RoCE and iWARP could be made more secure via existing tooling IPsec, and tcpcrypt respectively. We argue for the end-to-end principle and suggest that RDMA security should be TLS-like. The inclusion of encryption and authentication should only be on end hosts to prevent latency incursion at intermediate switches and routers. DTLS is the counterpart of TLS over UDP, and it works very similarly. Thus we only discuss a TLS-like implementation for TCP in the following paragraphs.

The TLS protocol ensures three properties: secrecy, authenticity, and reliability. Authenticity and reliability are less important in the datacenters. Every computing in-

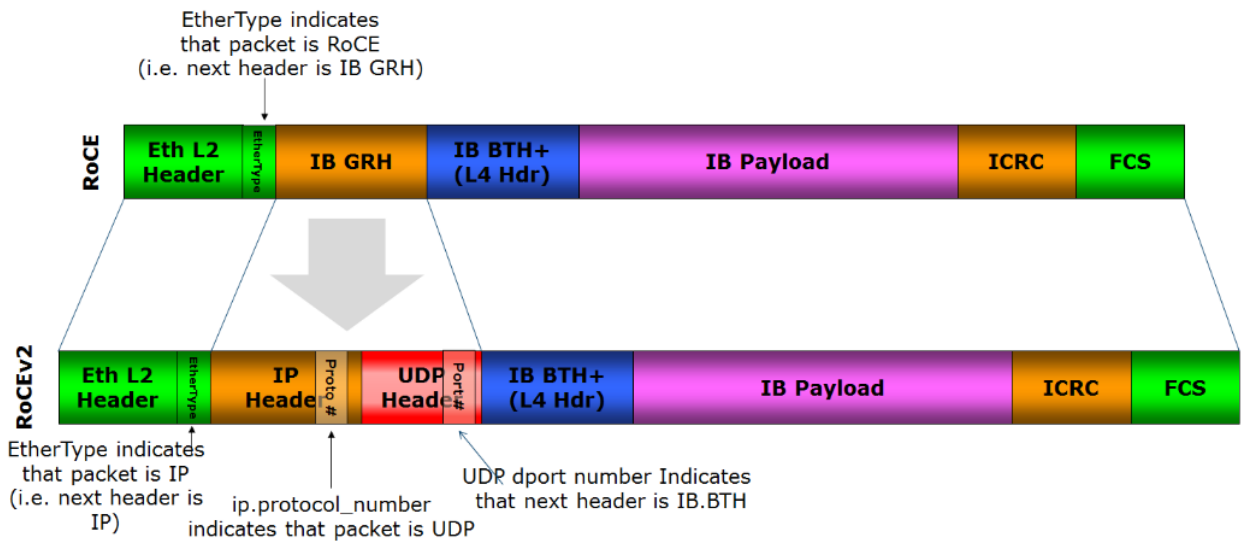

Figure 1: RoCE v1 and RoCE v2 packet format

stances in datacenters can be easily authenticated and we assume the networking in datacenter is reliable as lossless. Secrecy can be hard to achieve and we will present the difficulties at Sec. 5. A TLS implementation has two protocols: Handshake Protocols and Record Protocol. We argue that only Record Protocol is required in datacenters because the handshake protocols can be pre-assigned and updated in a fixed time period for a group of known hosts. In a RoCEv2 packet, there are one 4-byte CRC checksum for Ethernet FCS (Frame Check Sequence) and another 4-byye ICRC for RoCE v2 beyond Ethernet. Using a 4-byte MAC is not enough for any secure MAC algorithm for today's standard. For these reasons we argue that the MAC for TLS-enabled RDMA traffic should be implemented in the payload of InfiniBand.

There were very few works discussing RDMA security. A paper called Security Enhancement in InfiniBand Architecture [6] discussed the security aspect of an earlier version RDMA. They implemented authenticity by embedding an authentication tag in the header of an IBA packet.

4 Proof-of-Concept Attack

4.1 Test bed Setup

As proof of concept, and illustrative strawman for the weakness of RDMA we perform a man-in-the-middle attack. We demonstrate the power that an attacker with the ability to read and modify packets in the network can wield. For simplicity, and due to our lack of access to s specialized router that supports packet eavesdropping and manipulation, we used `scapy` to emulate router on a regular Ubuntu server.

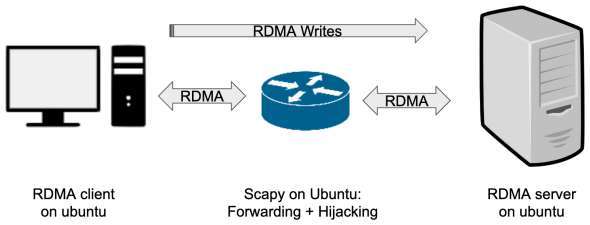

Figure 2: Attack Testbed Setup

We run a RoCEv2 RDMA client and server on two Ubuntu servers as seen in Figure Figure 2. Both client and server are connected to an attacker machine in the middle which uses `scapy` to sniff all `RoCEv2` packets, re-write the MAC addresses and forward it to an egress port. During attacks we allow the attacker to modify the packet as needed, after which we re-compute the `RoCEv2` checksum according to the IBTA specs [5], RoCE annex [3] and RoCEv2 annex [4].

We set up the `ARP` entries manually such that both client and server think they are talking to the each other directly, but in reality, the packets will all go through the middle machine. This configuration simulates a compromised switch in which an attacker has gained control over the data plane and can advertise arbitrary MAC address to connected hosts.

`scapy` is a `Python` module that allows packet sniffing, manipulation, injection, etc. It internally calls `tcpdump` and works on raw socket as well, which gives us the ability to re-write Ethernet-layer MAC addresses.

4.2 The RDMAttack

For our proof-of-concept attack, we choose to hijack the write address of RDMA write requests. Gaining control over a write location allows an attacker to place arbitrary data in a victims address space. Additionally it allows an attacker to seer RDMA traffic to select locations which can be used for Throwhammer exploits. We chose write as it gives the attacker explicitly more control over the victim. RDMA verbs for write are identical to reads, for the sake of brevity we could have modified read address and read sizes. This attack while omitted could be used to cause a victim to flood the network with Terabytes of unrequested data.

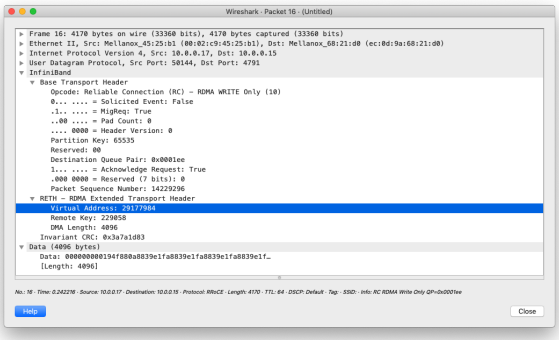

Figure 3: RDMA Write Packet in Wireshark

RDMA WRITE allows the requestor/client to write data to the address space of the target application on the target machine / server, using a remote key negotiated via prior authentication. Neither the key nor the virtual address is encrypted, as show in Figure 3.

We decided to hijack all writes to the same address, which can be handy to carry out Throwhammer attack as previously shown [7]. For our PoC, our adversary remembers the virtual address of the first WRITE it sees, and changes addresses of all subsequent WRITEs to be the same.

4.3 RDMA Client/Server

For simplicity, our RDMA client will issue n RDMA WRITE requests after initial handshake and information exchange needed for one-sided RDMA operations like RDMA WRITE. According to the attack model described in subsection 4.2, our attacker hijacks all subsequent RDMA WRITEs. Thus, we compare: 1) client issues on RDMA write, and 2) client issues two RDMA writes to two different address.

We used a modified RDMA performance benchmark tool, which in the first case, issues one RDMA WRITE of 16 bytes, and the second case, issues two consecutive RDMA WRITEs of 16 bytes each. At the end, both client and server computes the SHA1 checksum of the buffer region. If the SHA1 digest matches, then it means all data have been transferred successfully and not been tampered with.

4.4 Result

As show in Table 1, in the first case, there is only one RDMA WRITE, so the attacker doesn't modify the requests at all, thus the SHA1 digests match. But in the second case, since there are two RDMA writes, the second RDMA WRITE is hijacked so the buffer for the second WRITE did not complete successfully. Thus the SHA1 digests don't match. This simple proof-of-concept shows that it is (very) possible for attackers who compromised a switch to carry out RDMA attacks like Throwhammer and more.

5 Encryption at Line Rate: How Fast do we need?

The attack demonstrated in Sec. 4 suggests the necessity of encryption for RDMA traffic. However, applications using RDMA aim for high speed network links running at typically 40 to 100 Gbps. The RDMA traffic among machines is transferred in raw bytes without any encryption nowadays. This is an obvious but unconsidered trade-off between performance and security guarantees. There are, as we believe, more balanced solutions in the design space of a secure and performant RDMA system. In this section, we are going to discuss the encryption rate we need to support high-speed RDMA traffic as the first step toward it.

An ideal implementation for RDMA encrypted traffic should be very similar to TLS that implemented on top of the transport protocols. It should be able to operate at line-rate of the underlying network. As the link rate of network soars over the last few years, we can easily purchase 200 Gbps off-the-shelf network interface cards supporting RDMA. However, the encryption rate grows slower than this though having instruction-level support

Experiment #	1	2
SHA1 digest (client)	609f05f73fb251aec1c7d8b25b4cbe1d2d1a1661	b262a891b5dfc43930d6aa733e9741e625126a89
SHA1 digest (client)	609f05f73fb251aec1c7d8b25b4cbe1d2d1a1661	2488224f4d0dff339e01d1694a0bee162eb8c358

Table 1: RDMA Proof-of-Concept Result

on CPU, e.g. AES-NI on Intel x86 processors. We conducted a brief evaluation of encryption rate using common encryption algorithms and commodity CPUs. We chose to evaluate on block cipher, e.g. AES-CBC and AES-GCM and stream cipher, e.g. ChaCha20 with Poly1305. The evaluation executed on different block size ranging from 16 bytes to 8192 bytes. We ran encryption algorithms of OpenSSL 1.1.0 on a Intel Xeon E5-2650 v4 processor. It is single-threaded and with AES-NI enabled.

As Fig. 4 shown, the fastest AES-128-GCM encrypts 8k blocks around 26 Gbps which is way slower than the expected link rate of high speed network. Another downside of performing encryption on CPU is the waste of compute resources. CPUs are suppose to perform real computation instead of encryption. If we want to encrypt our data for RDMA traffic in 200 Gbps, then we need 8 physical cores dedicated for encryption. A server in datacenters usually has around 20 cores, and spending 8 cores dedicated for encryption is wasting 40% of the compute power.

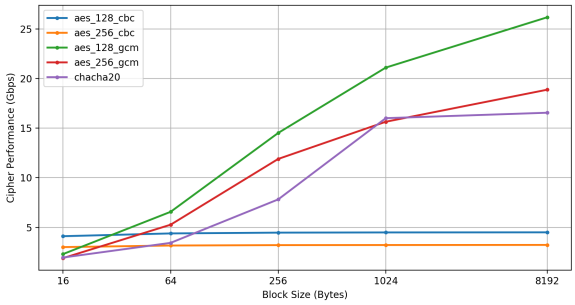

Figure 4: The encryption rate of an Intel CPU

The result shown in Fig. 4 demonstrates the infeasibility of using CPU for encryption. We believe that offloading encryption onto network interface card is necessary. Cavium LiquidIO network interface card operate at 25 Gbps with a single 25 Gbps SerDes device. It can encrypt data close to its 25 Gbps line rate as it shown in Cavium's datacenter security white paper. It supports major encryption and HMAC algorithms, e.g. AES-CBC for 128, 192 and 256 bits, AES-GCM for 128, 192 and 256 bits, HMAC-MD5, HMAC-AES-XCBC, HMAC-SHA1, HMAC-SHA for 256, 384, and 512 bits. As faster network interface cards, e.g. 100 Gbps and 200 Gbps, are consist of multiple 25 Gbps SerDes devices, we expect to observe similar encryption rate. In addition, Mellanox ConnectX-6 200 Gbps network interface card supports XTS-AES for 256 and 512 bits key. It targets storage traffic, but its serves as another strong evidence for encryption offloading on network interface cards.

6 Conclusion

In this work we demonstrated a critical fault in the security of RoCEv2, mainly that it has little to no security, and that a trivial man in the middle attack can recommend the protocol. In addition we benchmark the CPU performance of current encryption algorithms and show that any future RoCE standard must incorporate NIC offloaded encryption to support the line rate bandwidths demanded by RDMA.

References

[1] 802.1qbb priority-based flow control.

[2] BURNESS, E. Introducing the new hb and hc azure vm sizes for hpc, Sep 2018.

[3] INFINIBAND TRADE ASSOCIATION. *Supplement to InfiniBand Architecture Specification Volume 1 Release 1.2.1*, Apr. 2010. Annex A16: RDMA over Converged Ethernet (RoCE).

[4] INFINIBAND TRADE ASSOCIATION. *Supplement to InfiniBand Architecture Specification Volume 1 Release 1.2.1*, Sept. 2014. Annex A17: RoCEv2.

[5] INFINIBAND TRADE ASSOCIATION. *InfiniBand Architecture Specification Volume 1*, Mar. 2015. Release 1.3.

[6] LEE, M., KIM, E. J., AND YOUSIF, M. Security enhancement in infiniband architecture. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers - Volume 01* (Washington, DC, USA, 2005), IPDPS '05, IEEE Computer Society, pp. 105.1–.

[7] TATAR, A., KONOTH, R. K., ATHANASOPOULOS, E., GIUFFRIDA, C., BOS, H., AND RAZAVI, K. Throwhammer: Rowhammer attacks over the network and defenses. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)* (Boston, MA, 2018), USENIX Association, pp. 213–226.